Toward Real-time High-Frequency Stock Monitoring System using Node.js

Hao Qu[1], Kun Ma[1,2]*, Zhe Yang[1,3], Xuewei Niu[1], and Ajith Abraham[4]

[1] School of Information Science and Engineering, University of Jinan, Jinan 250022, China
houserqu@qq.com, ise_mak@ujn.edu.cn, v@yangzhe.net, niuxuewei@vip.qq.com
[2] Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China
[3] School of Computer Science and Technology, Shandong University, Jinan 250101, China
[4] Machines Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Auburn, WA 98071, USA
ajith.abraham@ieee.org

Abstract. Investing in stocks is a very popular financial management for modern people. Accurate data is aided for investor to make better decisions. But there are some challenges to design such a real-time system to present high-Frequent data. This paper introduced a solution toward real-time high-frequency stock monitoring system. The architecture, process and implementation of this system are presented respectively. This system is better in real-time monitoring, distributed services, high degree of concurrency, and interactive user experience using cache, load balancing, and CDN techniques. It is evaluated that Node.js performed well with advantages of its single thread event-driven mode and asynchronous I/O when it was in the environment of high concurrency.

Keywords: WebSocket, High Frequency, Stock Monitoring, Node.js, Socket.IO

1 Introduction

1.1 Background

With the establishment of Shanghai and Shenzhen exchange in China, the security exchange market has become globalized [1]. In the next decades, the scale of issuing market is increasing day by day, which lead the Chinese common people roll in stock market. The Internet plays an important part in the development of Chinese security exchange market. The high speed development of Internet provides convenient and rapid service for people, which makes the investor can trade securities online.

1.2 Motivation

Stock market is the integral part of security exchange market. On the demand on rapid economic development and online transaction demand, the situation puts forward higher requirement for the information construction. We should establish the effective restriction mechanism of information communication in the stock market to make sure users can get information more rapidly, more accurately and more conveniently. Because the stock market data is updated frequently, network delay is inevitable although it can be optimized by CDN (Content Delivery Network) [2] to a certain degree. And the bulk of the delay came from the remote name server. So the optimizing of the server and client is the main problem to be dealt with to ensure the effectiveness of the data. In addition, the accuracy of data is also important, so the system must have security and error handling mechanism.

1.3 Current Methods

Most of the stock software are client-server applications, such as Chinese DaZhi-Hui, TongHuaShun, and FuTuNN [3]. Users have to download and install the software packages if they want to use the software. Portability of them is poor. Once the software needs to be be updated, it needs to be downloaded and installed again. If you want to use the same software in other computers, you have to download it again. And there are variety of operating systems, software based on C/S structure need to be separately developed for every operating system, which increases the cost and time of development. Stock market applications based on B/S structure are relatively less and the solutions to show complicated data such as stock charts are not mature.

1.4 Contributions

Our project is based on B/S structure. It uses Node.js as JavaScript run-time system, Redis [4] as cache database, MySQL as persist storage database. In order to update data in the real-time way, we use WebSocket [5]. Firstly, the system gets the latest data continuously, stores them to the cache and then writes it into the database. While someone browses today's stock data, web page will update the data in time using WebSocket to realize the real-time display of stock data.

Applications based on B/S structure can be used across different platforms. First-time installation makes the software run anywhere, which reduces the cost and cycle of development. And the maintenance and upgrade operations will be simpler. It just need update system in servers. The clients need not to do anything. The whole process is just a moment for users. It also can save system resources of the users. Users don not need to download or install packages, they can finish every operation in browser.

The appearing of Node.js makes that JavaScript run on the server side. It is asynchronous I/O APIs to make it possible that it does not need to wait for result the until I/O operations finish. The main thread is in charge of issuing

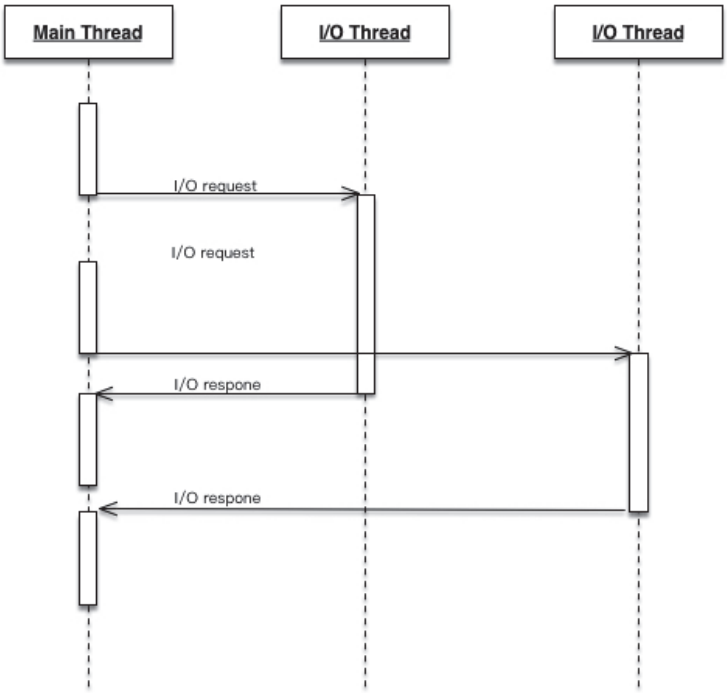

Fig. 1. Thread control of Node.js.

I/O request, and other I/O threads are in charge of executing I/O tasks and they return results after they finished. Main thread gets results from each I/O thread by event loop. In this way, it can process multiple requests in one thread rather than launch one thread for every request. So Node.js has the high efficiency of responding to requests. It consumes less system resources so it can use hardware resources more effectively and reduce the cost of maintenance. The Node.js thread control process is shown in Figure 1.

1.5 Organization

2 Related Work

2.1 Monitoring Techniques

WebSocket [6] is new communication protocol of HTML5, it realizes full-duplex between browser and server, replacing Polling and Comet technology. In this way it can auto update stock data real-timely and effectively by using lower server resource.

Redis [4] is a data storage base on memory. So it is very suitable for caching data. Stock information has strong timeliness. Providing a cache for frequently-accessed data might time-shift or reduce contention and overall request volumes to the database.

2.2 Monitoring Products

Nowadays there are many applications about stock market, such as Dazhuhui, Tonghuashun and so on [3]. Both of them are based on C/S structure, and they are developed only for Windows platform, which makes them have poor generality in the environment with multi-operation systems, they can only be used by part of users, and if the developers develop and maintain applications for every operation system, it will add significantly to the cost. The number of stock market applications which have perfect function and are based on B/S structure is very small. Obviously, although most of portals have stock market pages, such as Yahoo Finance, they mainly show stock market news and the form of the data is table or static picture, they are short on detail and intuitive.

Sina Finance Stock system [7], developed using PHP language, is based on B/S structure and it has realized main functions of stock, but its interfaces are old, which causes poor user experience. We tested it in different network environment, finding that the time for loading the whole page is average 5 seconds, and the time cost difference between the longest and lowest value is generally around 200ms, which shows that the process of updating data is unstable. This system is a kind of interpreted language like JavaScript. Apache is mainstream server software that run PHP, also it is for comparison in this paper.

3 Real-time High-frequency Stock Monitoring System

3.1 Architecture

Server frameworks that deal with highly concurrence are very mature now. As using distributed system, building cluster and using load balancing system. If hard resources are inadequate, they can add server to realize horizontal scale. Our system adopts distributed architecture as well, it architecture will be introduced in the following aspects. Back-end system is build on Node.js runtime environment, and it adopts Express [8] as web framework. Express provides a thin layer of fundamental web application features, rather than obscuring Node.js features. It is very minimal and flexible, so it can provide a robust set of features for web and mobile applications. In order to realize real time push function based on WebSocket, Socket.io [9] framework is adopted in the system to improve system compatibility and stability, which can realize variety of connections on different platforms.

The front-end adopts the newly HTML5 specification. In order to guarantee page performance, we used React framework developed by Facebook. Its

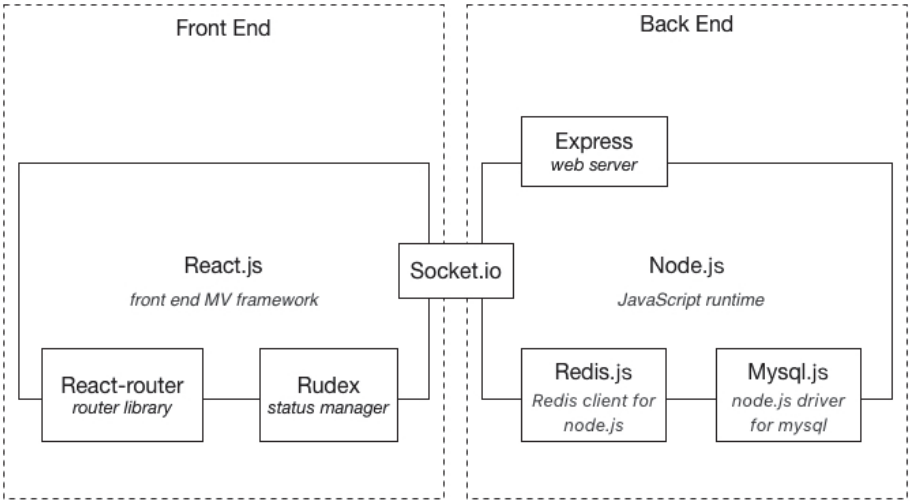

Fig. 2. Architecture of interactive developing.

component-based and virtual DOM improved the speed of browser render obviously. It also can be rendered on server, then the fluency of page will be further improved. The architecture of interactive developing is shown in Figure 2.

Database is always the bottleneck of the system performance, because all data read-write operations must be through database, but the data in database is saved on hard disk, its access speed is not so good. According to statistics, 80% of a web business concentrate on 20% data, the most accessed data of our system is stock data for the day. Therefore, we cache the data in memory in advance because memorys read speed is much faster than hard disk. In this way, it does not need to read data from database every time, so it can reduce the database access pressure, thus improve the whole web systems access speed.

Although the system uses the cache to make most of datas read operation finish in memory, in the case of cache miss, stale cache and executing writing operations, database is to be operated. If the scale of users is very large, database is still the bottleneck. MySQL has Master-salve replication function, it can sync one database data to other databases [10]. We can use this function to separate read operations and write operations. One server is only in charge of one operation, which can reduce the database load pressure. The architecture of database is shown in Figure 3.

If there are many users browsing the real-time stock information, the server need to build a larger number of socket connection, and the applications need much resources to keep this connection. Therefore, one server may not satisfy systems requirement. We can copy this system to multiple servers, and then use load-balancing dispatcher server to accept all uses requests and distribute them to different program servers. In this way, we can reduce single servers load and

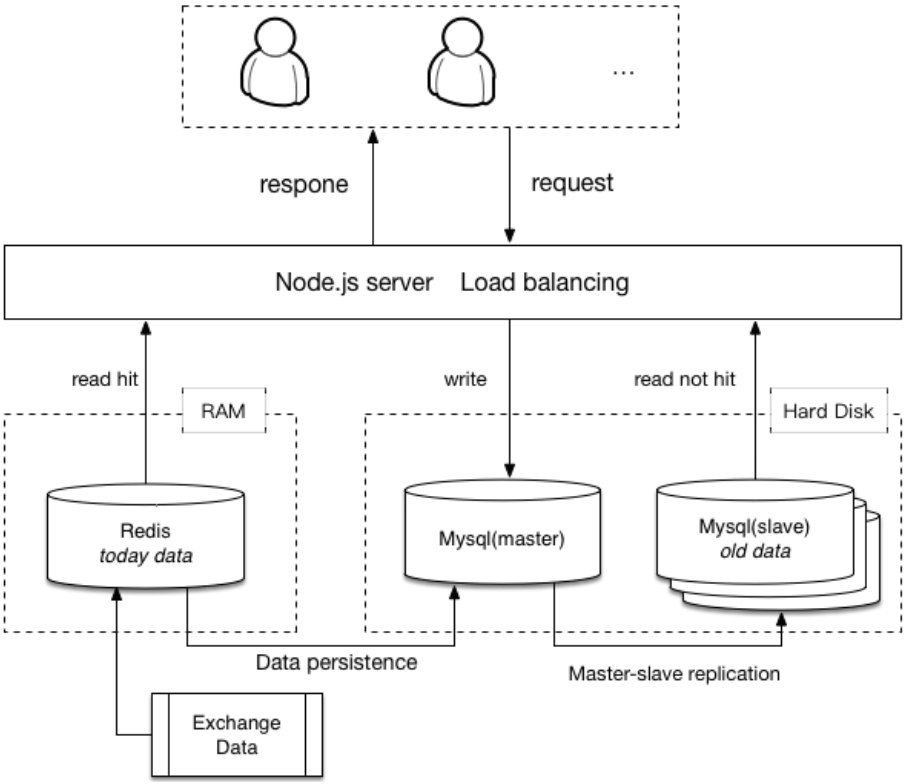

Fig. 3. Data architecture.

improve systems stability, which can make program server not be the bottleneck of the system. The architecture of load balancing is shown in Figure 4.

Users are from all over the world, and different users are in different Internet environment. If we put data of websites in same place, it will cause the users visit very slowly who are far from server network node, and stock information cant update timely even may cause the lost of information. On this occasion we can use CDN (Content Delivery Network), it can reallocate users request to the nearest node in real time on the basis of information such as network flow, connection and load condition, the distance to users and response time, in this way users can request stock information they need from the nearest node. The architecture of CDN is shown in the Figure 5.

3.2 Process

This system shown in Figure 6 does not generate data, but get the newest stock data from stock exchange or third-party APIs. Historical data can be written to

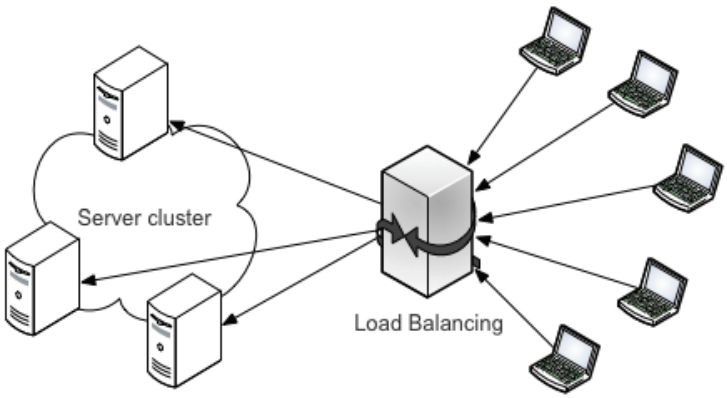

Fig. 4. Architecture of load balancing.

the database at once. The data include market index and stock index. System can show the trend of market index and stock index.

Users mainly browse stocks data for the day, and the data is refreshed by stock exchange every 2 seconds. System requests stock data offered by API every 2 seconds, then stores data to the memory of cache server and master database. Master database copy data to slave database by master-slave replication.

When users enter websites to browse the real-time information for the day, first the load balancing will choose suitable server by the servers loading condition to response users request, then read related data generated that day and shape a chart displayed on the website. If users do not close the page, once the data offered by API change, data will be written to cache server and database, then application server will read the newest data from cache server. Finally the data will be pushed on users browser by Socket.io, and the stock information refresh automatically. When users browse historical stock information, data will be read from database directly. Every time stock market open, yesterdays data will be cleared, and the space left will be used for todays stock data.

3.3 Implementation

This System mainly offer convenient and real-time stock market information. The function modules of the whole system are as below

- Account manager. Every user has its own data, so system should provide account manage function to record their personal data, such as login information and stock collection.
- Displaying information of the stock market.
  - The list of stock market. We can see all of the real-time detailed information of Hongkong stock market, Shenzhen stock market, America stock market and Shanghai stock market. Every stocks data include: code,

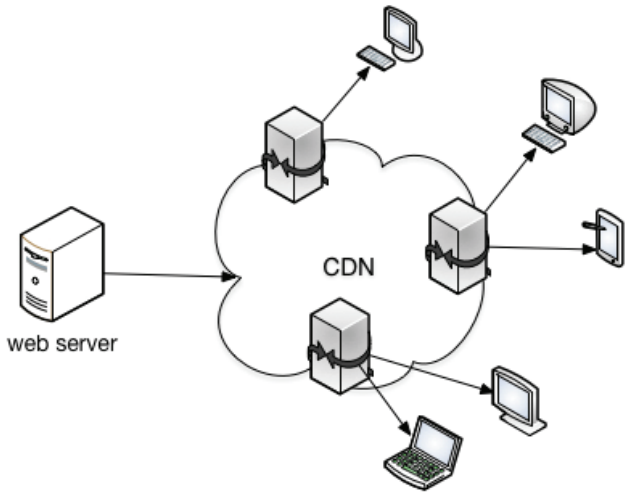

Fig. 5. Architecture of CDN.

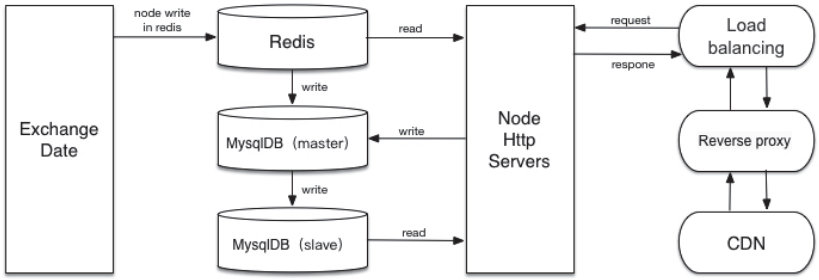

Fig. 6. System architecture.

name, English name, last price, open, high, low, volume, turnover time, buy, sell, 52 week, change amount, change rate.
- The broad market: Trend chart and K-chart.
- Stock searching.

– Adding user stock. Let user see his favorite stock expediently.

4 Experimental results

In this section, we evaluate the performance of Real-time High-Frequency Stock Monitoring System using Node.js compared with Apache.

Our system uses Node.js engine, whose model is to use a main thread to manage all request and get result by asynchronous I/O requests. Figure 7 is the memory usage of Node.js and Apache in the same hardware configuration

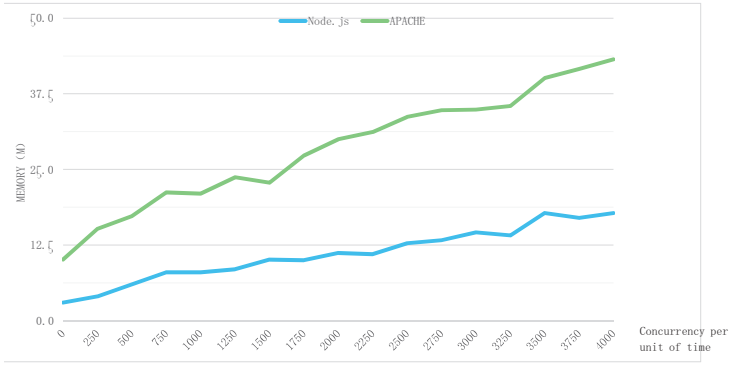

Fig. 7. Memory usage of Node.js and Apache.

(CPU: 2.7 GHz Intel Core i5, RAM: 8 GB 1867 MHz DDR3) and concurrence. It indicates that Node.js memory dissipation is lower than Apache.

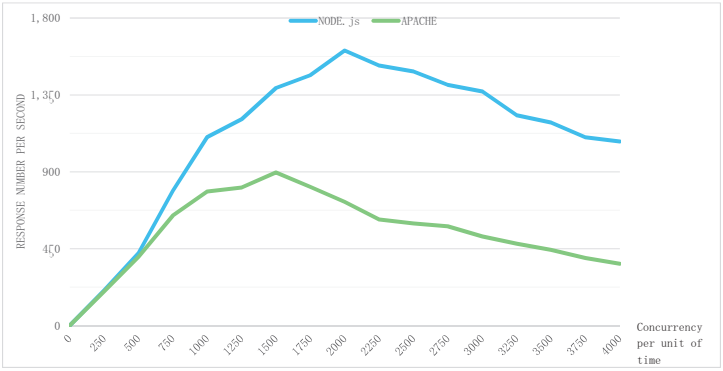

Fig. 8. Response of Node.js and Apache with the increase of concurrency per unit of time.

Figure 8 is the response of Node.js and Apache with the increase of concurrency per unit of time in the same hardware configuration (CPU: 2.7 GHz Intel Core i5, RAM: 8 GB 1867 MHz DDR3) and concurrence. Node.js also has enormous advantages in response rate.

5 Conclusions

With the development of Internet, the accurate and timely access of the information are becoming more and more important, and people's demand for Internet

are becoming stronger and stronger. At the same time, the Internet technology is flourishing, for example, JavaScript is a script language which could be run only on browser in the past, now it can be run on the server with the help of Node.js environment. This paper introduced a solution of toward real-time high-frequency stock monitoring system. It is proved that Node.js performed well with advantages of its single thread event-driven mode and asynchronous I/O when it was in the environment of high concurrency. This system is suitable for distributed and data-intensive business, meeting high concurrence with little hardware resources. This solution is a new technology to achieve better effect performance of monitoring systems.

Acknowledgments. This work was supported by the Shandong Provincial Natural Science Foundation (ZR2014FQ029), the Shandong Provincial Key R&D Program (2015GGX106007), the Open Project Funding of Shandong Provincial Key Laboratory of Software Engineering (2015SE03), and the Project of Shandong Province Higher Educational Science and Technology Program (J16LN13). Dr. Kun Ma is the corresponding author.

References

1. Chang, E.C., Luo, Y., Ren, J.: Short-selling, margin-trading, and price efficiency: Evidence from the chinese market. Journal of Banking & Finance **48** (2014) 411–424
2. Spagna, S., Liebsch, M., Baldessari, R., Niccolini, S., Schmid, S., Garroppo, R., Ozawa, K., Awano, J.: Design principles of an operator-owned highly distributed content delivery network. IEEE Communications Magazine **51**(4) (2013) 132–140
3. Hu, Y., Zhang, X., Feng, B., Xie, K., Liu, M.: itrade: A mobile data-driven stock trading system with concept drift adaptation. International Journal of Data Warehousing and Mining (IJDWM) **11**(1) (2015) 66–83
4. Pokorny, J.: Nosql databases: a step to database scalability in web environment. International Journal of Web Information Systems **9**(1) (2013) 69–82
5. Ma, K., Zhang, W.: Introducing browser–based high–frequency cloud monitoring system using websocket proxy. International Journal of Grid and Utility Computing **6**(1) (2014) 21–29
6. Ma, K., Sun, R.: Introducing websocket-based real-time monitoring system for remote intelligent buildings. International Journal of Distributed Sensor Networks **2013** (2013)
7. Wu, D.D., Olson, D.L.: Online stock forum sentiment analysis. In: Enterprise Risk Management in Finance. Springer (2015) 49–56
8. Ra, H.K., Yoon, H.J., Salekin, A., Lee, J.H., Stankovic, J.A., Son, S.H.: Software architecture for efficiently designing cloud applications using node. js. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion, ACM (2016) 72–72
9. Rai, R.: Socket. IO Real-time Web Application Development. Packt Publishing Ltd (2013)
10. Watson, M.: Scripting intelligence: Web 3.0 information gathering and processing. Apress (2009)